School of Computing Science & Engineering (SCOPE)

Practical File

Course Name: Programming in Java
(Course Code: CSE2006)
Slot: C12 + C13

Submitted By:
Name: LAKSHYA KANT
Reg. no.: 24BCE10549

Submitted to:
Dr. Sanat Jain

PROGRAMMING IN JAVA
CSE2006

INDEX

Exp. No.	Title	Date Conducted	Remark
1	Java Program to Check whether input character is vowel or consonant.	04-07-2025	
2	Java Program to Find Factorial of a number.	09-07-2025	
3	Java Program to Show Encapsulation in Class.	11-07-2025	
4	Java Program to Show Inheritance in Class.	16-07-2025	
5	Java Program to Show Polymorphism in Class.	18-07-2025	
6	Java Program to Show Overloading of Methods in Class.	23-07-2025	
7	Java Program to Show Overriding of Methods in Classes.	25-07-2025	
8	Java Program to Show Use of Super Keyword in Class.	30-07-2025	
9	Java Program to Show Use of This Keyword in Class.	01-08-2025	
10	Java Program to Show Usage of Static keyword in Class.	02-08-2025	
11	Java Program to Show Usage of Access Modifier.	06-08-2025	
12	Java Program to demonstrate constructor overloading.	13-08-2025	
13	Java Program to Create Abstract Class.	29-08-2025	
14	Java Program to Create an Interface.	03-09-2025	
15	Java Program to Create Singleton Class.	05-09-2025	

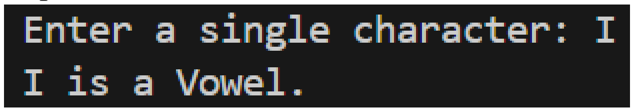
EXPERIMENT 1:

Aim: Write a Java Program to Check whether input character is vowel or consonant.

Code:

```java
import java.util.Scanner;

public class VowelOrConsonant {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a single character: ");
        char ch = sc.next().charAt(0);

        char[] vowels = {'a','e','i','o','u','A','E','I','O','U'};
        boolean isVowel = false;

        for (char v : vowels) {
            if (ch == v) {
                isVowel = true;
                break;
            }
        }

        if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
            if (isVowel)
                System.out.println(ch + " is a Vowel.");
            else
                System.out.println(ch + " is a Consonant.");
        } else {
            System.out.println("Invalid input. Please enter an
alphabet.");
        }

        sc.close();
    }
}
```

Output:



```
Enter a single character: I
I is a Vowel.
```

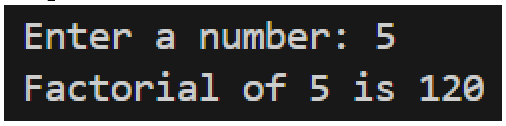
EXPERIMENT 2:

Aim: Write a Java Program to Find Factorial of a number.

Code:

```java
import java.util.Scanner;

public class FactorialRecursive {
    static long factorial(int n) {
        if (n == 0 || n == 1)
            return 1;
        return n * factorial(n - 1);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        if (num < 0)
            System.out.println("Factorial is not defined for negative
numbers.");
        else
            System.out.println("Factorial of " + num + " is " +
factorial(num));
        sc.close();
    }
}
```

Output:



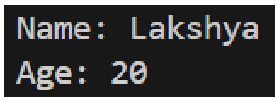
```
Enter a number: 5
Factorial of 5 is 120
```

EXPERIMENT 3:

Aim: Java Program to Show Encapsulation in Class.

Code:

```
class Student {
    private String name;
    private int age;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        if (age > 0)
            this.age = age;
        else
            System.out.println("Age must be positive.");
    }

    public int getAge() {
        return age;
    }
}

public class EncapsulationDemo {
    public static void main(String[] args) {
        Student s = new Student();
        s.setName("Lakshya");
        s.setAge(20);

        System.out.println("Name: " + s.getName());
        System.out.println("Age: " + s.getAge());
    }
}
```

Output: 

```
Name: Lakshya
Age: 20
```

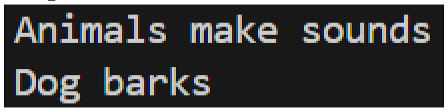
EXPERIMENT 4:

Aim: Java Program to Show Inheritance in Class.

Code:

```java
class Animal {
    void sound() {
        System.out.println("Animals make sounds");
    }
}

class Dog extends Animal {
    void sound() {
        System.out.println("Dog barks");
    }
}

public class InheritanceDemo {
    public static void main(String[] args) {
        Animal a = new Animal();
        a.sound();

        Dog d = new Dog();
        d.sound();
    }
}
```

Output:



```
Animals make sounds
Dog barks
```

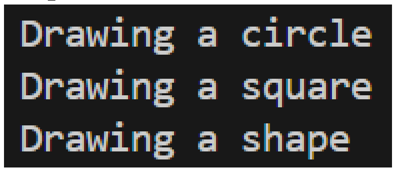
EXPERIMENT 5:

Aim: Java Program to Show Polymorphism in Class.

Code:

```java
class Shape {
    void draw() {
        System.out.println("Drawing a shape");
    }
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a circle");
    }
}

class Square extends Shape {
    void draw() {
        System.out.println("Drawing a square");
    }
}

public class PolymorphismDemo {
    public static void main(String[] args) {
        Shape s1 = new Circle();
        Shape s2 = new Square();
        Shape s3 = new Shape();

        s1.draw();
        s2.draw();
        s3.draw();
    }
}
```

Output:



```
Drawing a circle
Drawing a square
Drawing a shape
```

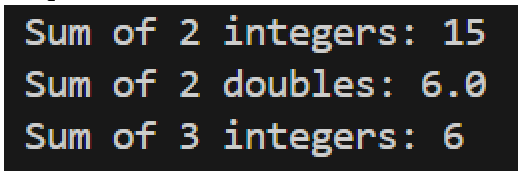
EXPERIMENT 6:

Aim: Java Program to Show Overloading of Methods in Class.

Code:

```java
class Calculator {
    int add(int a, int b) {
        return a + b;
    }

    double add(double a, double b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }
}

public class OverloadingDemo {
    public static void main(String[] args) {
        Calculator calc = new Calculator();

        System.out.println("Sum of 2 integers: " + calc.add(5, 10));
        System.out.println("Sum of 2 doubles: " + calc.add(3.5, 2.5));
        System.out.println("Sum of 3 integers: " + calc.add(1, 2, 3));
    }
}
```

Output:



```
Sum of 2 integers: 15
Sum of 2 doubles: 6.0
Sum of 3 integers: 6
```

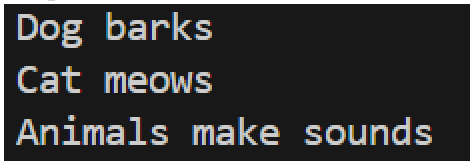
EXPERIMENT 7:

Aim: Java Program to Show Overriding of Methods in Classes.

Code:

```java
class Animal {
    void sound() {
        System.out.println("Animals make sounds");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

public class OverridingDemo {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        Animal a2 = new Cat();
        Animal a3 = new Animal();

        a1.sound();
        a2.sound();
        a3.sound();
    }
}
```

Output:

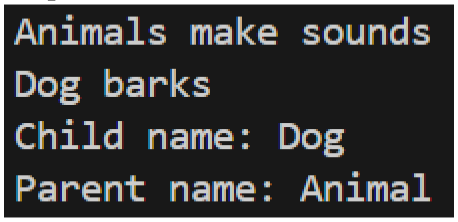


```
Dog barks
Cat meows
Animals make sounds
```

EXPERIMENT 8:

Aim: Java Program to Show Use of Super Keyword in Class

Code:
```java
class Animal {
    String name = "Animal";

    void sound() {
        System.out.println("Animals make sounds");
    }
}

class Dog extends Animal {
    String name = "Dog";

    void sound() {
        super.sound();
        System.out.println("Dog barks");
    }

    void showNames() {
        System.out.println("Child name: " + name);
        System.out.println("Parent name: " + super.name);
    }
}

public class SuperKeywordDemo {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.sound();
        d.showNames();
    }
}
```

Output:

```
Animals make sounds
Dog barks
Child name: Dog
Parent name: Animal
```

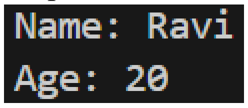
EXPERIMENT 9:

Aim: Java Program to Show Use of This Keyword in Class.

Code:

```java
class Student {
    String name;
    int age;

    Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + this.name);
        System.out.println("Age: " + this.age);
    }

    void show() {
        this.display();
    }
}

public class ThisKeywordDemo {
    public static void main(String[] args) {
        Student s = new Student("Ravi", 20);
        s.show();
    }
}
```

Output:



```
Name: Ravi
Age: 20
```

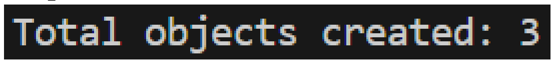
EXPERIMENT 10:

Aim: Java Program to Show Usage of Static keyword in Class.

Code:

```java
class Counter {
    static int count = 0;

    Counter() {
        count++;
    }

    static void displayCount() {
        System.out.println("Total objects created: " + count);
    }
}

public class StaticKeywordDemo {
    public static void main(String[] args) {
        Counter c1 = new Counter();
        Counter c2 = new Counter();
        Counter c3 = new Counter();

        Counter.displayCount();
    }
}
```

Output:



```
Total objects created: 3
```

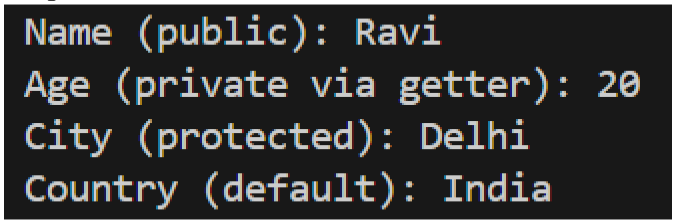
EXPERIMENT 11:

Aim: Java Program to Show Usage of Access Modifier.

Code:

```
class Person {
    public String name = "Ravi";          // Public: accessible everywhere
    private int age = 20;                 // Private: accessible only
within this class
    protected String city = "Delhi";      // Protected: accessible within
same package & subclasses
    String country = "India";             // Default: accessible only
within same package

    public int getAge() {
        return age; // private accessed via getter
    }
}

public class AccessModifierDemo {
    public static void main(String[] args) {
        Person p = new Person();

        System.out.println("Name (public): " + p.name);
        System.out.println("Age (private via getter): " + p.getAge());
        System.out.println("City (protected): " + p.city);
        System.out.println("Country (default): " + p.country);
    }
}
```

Output:



```
Name (public): Ravi
Age (private via getter): 20
City (protected): Delhi
Country (default): India
```

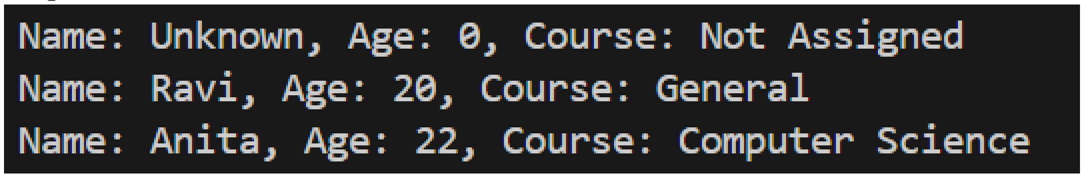
EXPERIMENT 12:

Aim: Java Program to demonstrate constructor overloading.

Code:

```java
class Student {
    String name;
    int age;
    String course;

    Student() {
        name = "Unknown";
        age = 0;
        course = "Not Assigned";
    }

    Student(String name, int age) {
        this.name = name;
        this.age = age;
        this.course = "General";
    }

    Student(String name, int age, String course) {
        this.name = name;
        this.age = age;
        this.course = course;
    }

    void display() {
        System.out.println("Name: " + name + ", Age: " + age + ",
Course: " + course);
    }
}

public class ConstructorOverloadingDemo {
    public static void main(String[] args) {
        Student s1 = new Student();
        Student s2 = new Student("Ravi", 20);
        Student s3 = new Student("Anita", 22, "Computer Science");

        s1.display();
        s2.display();
        s3.display(); }}
```

Output :



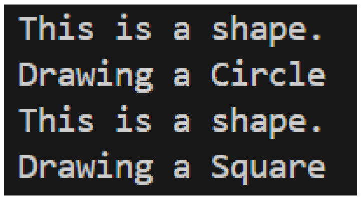
```
Name: Unknown, Age: 0, Course: Not Assigned
Name: Ravi, Age: 20, Course: General
Name: Anita, Age: 22, Course: Computer Science
```

EXPERIMENT 13:

Aim: Java Program to Create Abstract Class.

Code:

```java
abstract class Shape {
    abstract void draw();

    void info() {
        System.out.println("This is a shape.");
    }
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a Circle");
    }
}

class Square extends Shape {
    void draw() {
        System.out.println("Drawing a Square");
    }
}

public class AbstractClassDemo {
    public static void main(String[] args) {
        Shape s1 = new Circle();
        Shape s2 = new Square();

        s1.info();
        s1.draw();

        s2.info();
        s2.draw();
    }
}
```

Output: 

```
This is a shape.
Drawing a Circle
This is a shape.
Drawing a Square
```

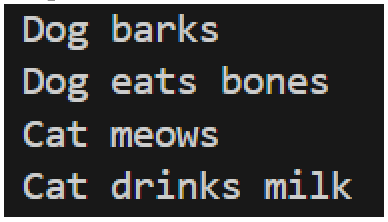
EXPERIMENT 14:

Aim: Java Program to Create an Interface.

Code:

```java
interface Animal {
    void sound();
    void eat();
}

class Dog implements Animal {
    public void sound() {
        System.out.println("Dog barks");
    }

    public void eat() {
        System.out.println("Dog eats bones");
    }
}

class Cat implements Animal {
    public void sound() {
        System.out.println("Cat meows");
    }

    public void eat() {
        System.out.println("Cat drinks milk");
    }
}

public class InterfaceDemo {
    public static void main(String[] args) {
        Animal a1 = new Dog();
        Animal a2 = new Cat();

        a1.sound();
        a1.eat();

        a2.sound();
        a2.eat();
    }
}
```

Output:



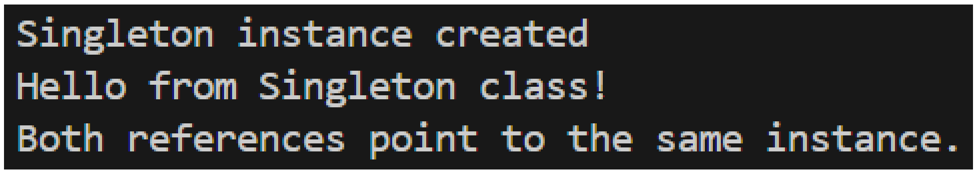
```
Dog barks
Dog eats bones
Cat meows
Cat drinks milk
```

EXPERIMENT 15:

Aim: Java Program to Create Singleton Class.

Code:

```java
class Singleton {
    private static Singleton instance;

    private Singleton() {
        System.out.println("Singleton instance created");
    }

    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }

    public void showMessage() {
        System.out.println("Hello from Singleton class!");
    }
}

public class SingletonDemo {
    public static void main(String[] args) {
        Singleton obj1 = Singleton.getInstance();
        Singleton obj2 = Singleton.getInstance();

        obj1.showMessage();

        if (obj1 == obj2) {
            System.out.println("Both references point to the same 
instance.");
        }
    }
}
```

Output: 

```
Singleton instance created
Hello from Singleton class!
Both references point to the same instance.
```